## ARTIFICIAL INTELLIGENCE FINAL PROJECT

On

RL-BASED SYSTEM FOR ASSISTING CAB DRIVERS

Submitted By

**Chakra Mohan Kumar Kota**

**Krishna Rao Kundeti**

## Under the Guidance of

**VAHID BEHZADAN** (Assistant Professor)



# Project Objective:-

Reinforcement learning based system is for assisting the cab drivers which can enables a driver, to choose the rides which are likely to optimize the total profit earned by him at end of the day and daily profits by improving for their decision-making process on the field.

In this highly aggressive industry, detention of good cab drivers is more significant in business. Cab drivers will like most people who are boosted up by their healthy growth in their income. But with the recent ramble in electricity prices (all cabs are electric), many drivers complain that although their incomes are gradually increasing but their profits are almost unbroken. Thus, it is important that drivers choose for their 'right' rides for example, let us suppose say that a driver gets three ride requests at 11 PM. The first one is a long-distance ride assure high fare, but it will take him to a location which is unlikely to get him another ride for the next few hours. The second one ends in a better location, but it requires him to take a slight diversion to pick the customer up, adding to fuel costs. Perhaps the best choice is to choose the third one, which although is medium distance, it will likely get him another ride subsequently and avoid most of the traffic.

# APPROACH: -

In our project, that we need to create an environment and an RL agent that will learn to choose the request that We need to train our agent using Deep Q-learning (DQN).

##### Goals: -

**Create the environment:**

The ‘Env.py’ file is the "environment class" - each method (function) of the class has a specific purpose.

##### Build an Agent:

Building an agent that will learns to pick the best request using DQN that We can choose the hyperparameters (epsilon (go off rate), learning-rate etc.) of our choice.

Training totally depends purely on the epsilon-function that we choose. If it decays fast, it won’t let our model explore as much and the Q-values will converge early but to suboptimal values. If it decays slowly, our model will converge slowly.

# Assumptions: -

1. The taxis are electric cars. It can run for 30 days non-stop, i.e., 24\*30 hours. Then it needs to recharge itself. If the cab driver is completing his trip at that time, he will finish that trip and then stop for recharging.
2. All decisions will be made at hourly intervals. We won’t consider minutes and seconds for this project. So, for example, suppose the cab driver gets request at 10 PM then at 9PM and so on. He can decide to pick among the requests only at these times. A request cannot come at 9.30 PM.
3. The time taken to travel from one place to another is considered in integer hours only and is dependent on the traffic. Also, the traffic is dependent on the hour-of-the-day and the day-of-the- week.

### Reinforcement Learning: -

Diagram

Description automatically generated

Reinforcement Learning is a subset of machine learning. It also enables an agent to learn through the consequences of an actions in a specific environment and also it is the area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of reward.

It also differs from other forms of supervised learning because the sample data set does not train the machine. Instead, it learns by trial and error. Therefore, a series of right decisions would be the method as it better solves the problem.

In here RL is well known as semi-supervised model in machine learning and it is alsoa technique to be allowed as an agent to take actions and interact with an environment so as to maximize the total rewards.

Diagram

Description automatically generated

. Applications areas of RL: -

* + Personalized Recommendations
  + Games
  + Deep Learning
  + Robotics
  + Business
  + Manufacturing
  + Finance sectors

Diagram

Description automatically generated

##### Challenges: -

* + Reinforcement learning’s key challenge is to plan the simulation environment, which relies heavily on the task to be performed.
  + Transferring the model from the training setting to the real world becomes problematic.
  + Scaling and modifying the agent’s neural network is another problem.

Reinforcement is done with rewards according to the decisions made; it is possible to always learn continuously from interactions with the environment. With each correct action, we will have positive rewards and penalties for incorrect decisions. In the industry, this type of learning can help optimize processes, simulations, monitoring, maintenance, and the control of autonomous systems.

##### Markov Decision Process:-

In mathematics, a Markov decision process (MDP) is a discrete-time theoretical control process. It provides a mathematical framework for modeling decision making in situations where their outcomes are partly random and partly under the control of a decision maker.

A Markov decision process is a 4-tuple **(S, A, P\_a, R\_a**), where:

* S is a set of states called the state space,
* A is a set of actions called the action space (alternatively, A\_s is the set of actions available from state s),
* P(s, s')= Pr(s\_t+1=s'|s\_t=s, a\_t=a) is the probability that action a in state s at time t will lead to state s' at time t+1,
* R\_a(s, s') is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a

The state and action spaces may be finite or infinite, for example the set of real numbers. Some processes with countably infinite state and action spaces can be reduced to ones with finite state and action spaces.

A policy function pi is a (potentially probabilistic) mapping from state space to action space.

### Deep Q-Learning: -

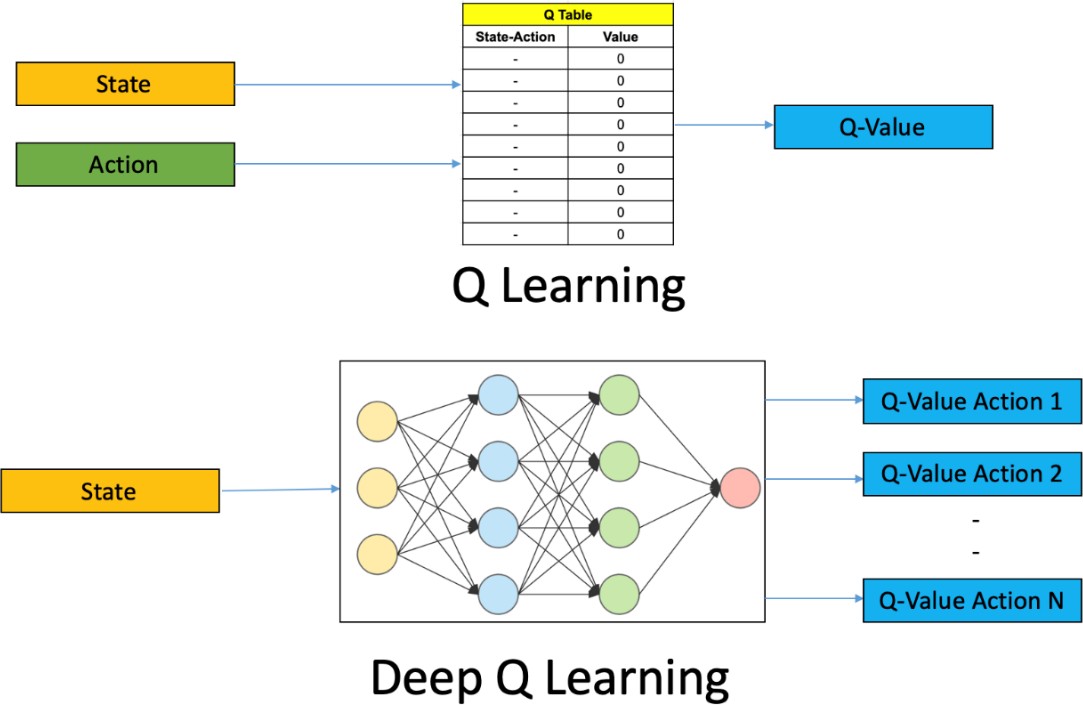
### Diagram Description automatically generated

Q-learning is a simple yet quite powerful algorithm to create a cheat sheet for our agent. This helps the agent to figure out exactly which action to be performed.

This presents two problems:

* + Here in First, the amount of memory required to save and update that table would increase as the number of states increases.
  + Second, the amount of time required to explore each state to create the required Q-table would be unrealistic.

In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output. The comparison between Q-learning & deep Q-learning is wonderfully illustrated as below.



Code Explanation:-

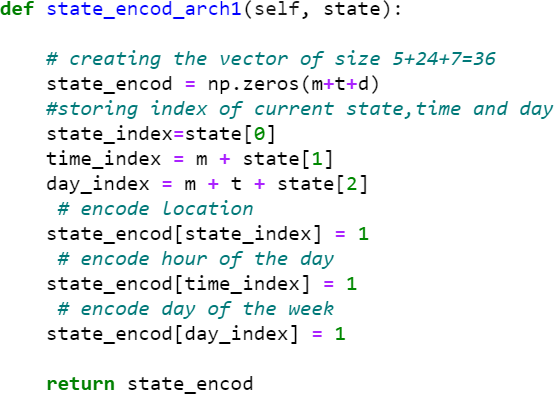
#### Environment Class: -

A reinforcement learning task is about training an agent which interacts with its environment. The agent arrives at different scenarios known as states by performing actions. Actions lead to rewards which could be positive and negative.

The agent has only one purpose here – to maximize its total reward across an episode. This episode is anything and everything that happens between the first state and the last or terminal state within the environment. We reinforce the agent to learn to perform the best actions by experience. This is the strategy or policy.

Graphical user interface, text, application

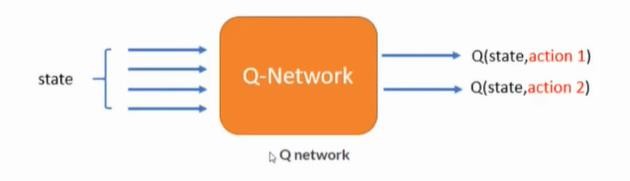
Description automatically generated



We have 2 architectures of DQN,

1. We pass only State as input
2. We pass State and Action as input

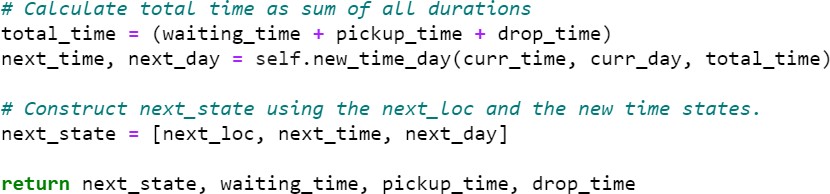
The Architecture 1 (only State as input) performs better than Architecture 2 because we will get Q(s, a) for each action, so you have to run the NN just once for every state. Take the action for which Q(s, a) is the maximum.



#### Next State function: -

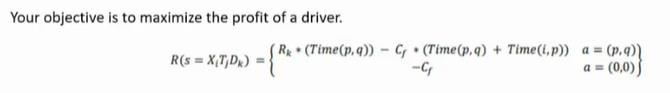
Takes state and action as input and returns next state with considering below conditions.

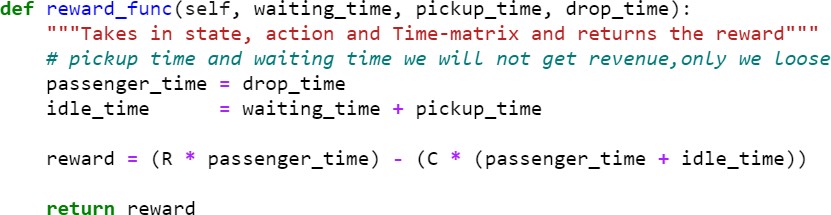
1. driver refuse to request.
2. cab is already at pick up point.
3. cab is not at the pickup point.



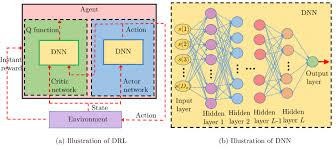
## Reward function: -

Assessment requires to determine what action is to be taken to minimize loss and maximize benefits. The reward, r(s, a), in our system for taking an action a ∈ A at a given state s ∈ S is computed as follows.





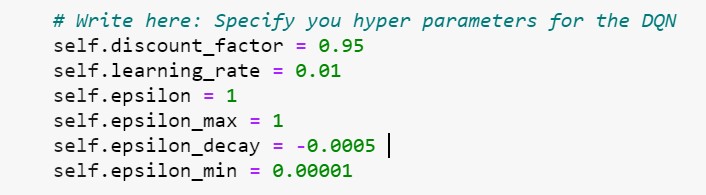
## Cab Driver DQN Agent: -



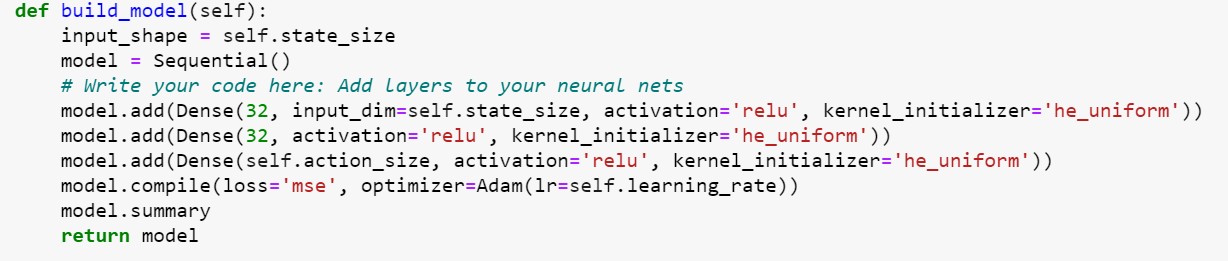
In Agent class we need to work on below functions are

* + Assigning hyperparameters
  + Creating a neural-network model.
  + Define epsilon-greedy strategy.
  + Appends the recent experience state, action, reward, new state to the memory.
  + Build the DQN model using the updated input and output batch. Hyperparameters: -

We can tweak these parameters for better performance.



Neural Network Model: -



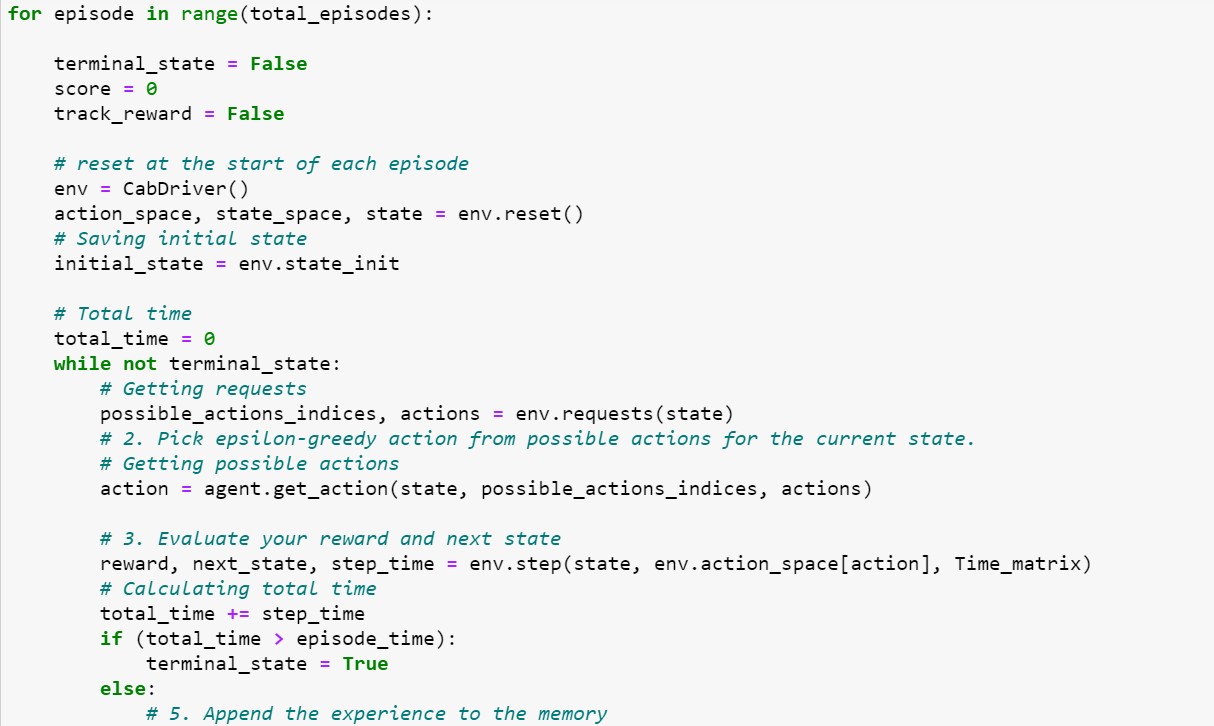
Using keras we build a sequential model by adding dense layers.

We have provided state as input at the first layer with relu nonlinear activation function and then added hidden layers for better learning with relu activation function.

Here, we are using Adam optimizer which uses epsilon greedy policy and learning rate to improve the weights and bias to minimize mean square error.

Build the DQN model: -

Appends the recent experience state, action, reward, new state to the memory with updated input and output batch.



Graphical user interface, text, application, chat or text message

Description automatically generated

# Evaluation : -

Model continuously update strategies to learn a strategy that maximizes long-term cumulative rewards.

Below two are the performance matrices for our model.

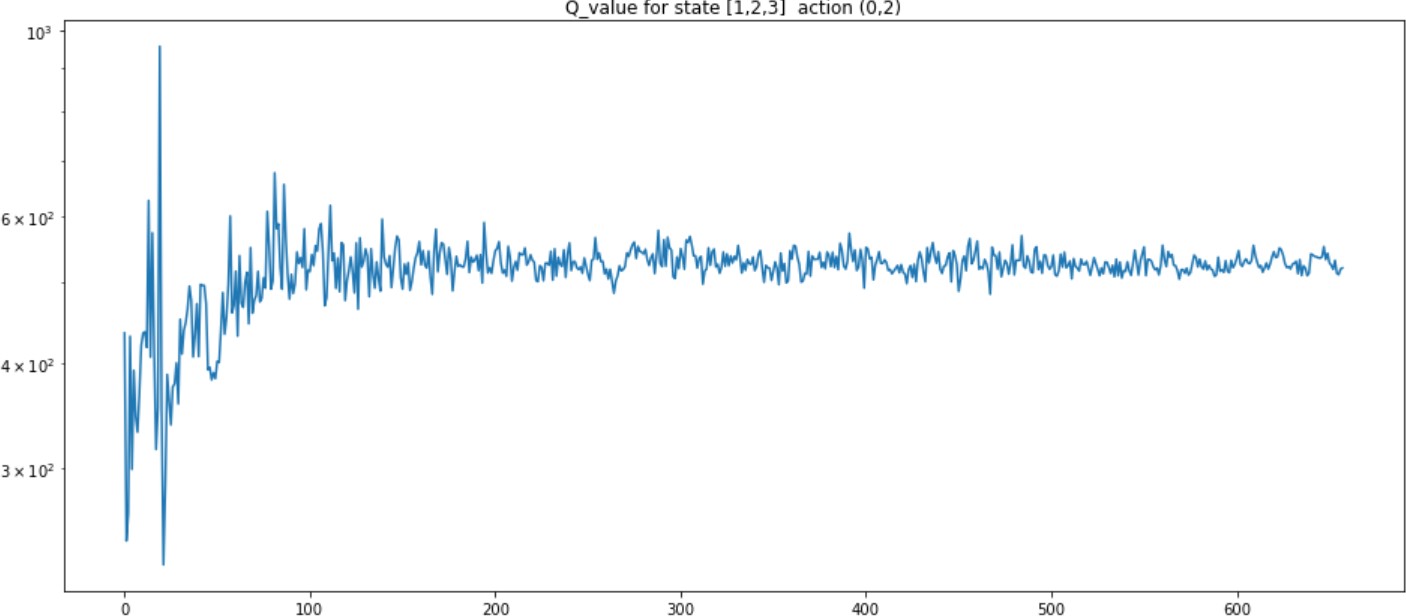
* Q-Value convergence.
* Rewards per episode. Text

  Description automatically generated

##### After few episodes, our agent learns to choose best request with an experience and provides better rewards.

Text

Description automatically generated

**Q- Value Convergence**Graphical user interface, chart

Description automatically generated

Graphical user interface, text

Description automatically generated

# Conclusion: -

We have modeled a RL-Based system agent using Deep Q-Learning Network, which can help cab drivers to maximize their profits by improving agent’s decision-making process. We have plotted Q-value convergence and rewards per episode to understand the model performance. As we increase the number of rides in the system, the collected reward increases.

# References: -

<https://arxiv.org/abs/2104.12000> <https://arxiv.org/abs/2104.12226>

<https://towardsdatascience.com/about-reinforcement-learning-2ff0dafe9b75>